УДК 004.62 /.652/.657

А.Г. Морозова, Л.П. Белова, В.А. Погребняк

Харьковский национальный университет имени В.Н. Каразина, Харьков

СПЕЦИФИКАЦИЯ ОГРАНИЧЕНИЙ ЦЕЛОСТНОСТИ ДЛЯ NOSQL БАЗ ДАННЫХ

В работе предложен метод описания ограничений целостности для NoSQL баз данных. Рассмотрены основные модели NoSQL баз данных, а также целесообразность описания ограничений целостности для каждой из моделей. Для описания ограничений целостности было предложено описать каждую NoSQL модель на формальном языке – SDL в рамках теории предсхем, что позволило сформулировать ограничения целостности на формальном языке, основанном на пути в ациклическом графе. В рамках этого подхода разработан алгоритм проверки ограничений целостности для основных моделей NoSQL баз данных.

Ключевые слова: *базы данных, модели данных, NoSQL, распределенные базы данных, горизонтальное масштабирование, хранилище ключ-значение, документно-ориентированные базы данных, хранилища семейств колонок, графовые базы данных.*

Введение

С развитием интернет технологий и огромным ростом информации возникла проблема ее обработки за короткое время, а также проблема горизонтального масштабирования. Традиционные реляционные СУБД не в состоянии были решить эти проблемы, что привело к появлению распределенных информационных систем [1].

Распределение нагрузки позволило увеличивать мощность компьютерных систем через горизонтальное масштабирование, которое является во многих случаях более предпочтительным, нежели вертикальное. Несмотря на свои плюсы, горизонтальное масштабирование сопряжено с изменением структуры процессов, проходящих в системе, и накладывает определенные ограничения, связанные с распределенностью вычислительных ресурсов и хранилищ данных.

Рост числа распределенных информационных систем обусловил необходимость в создании новых «нереляционных» баз данных – или NoSQL баз данных. NoSQL базы данных – термин, обозначающий ряд подходов, направленных на реализацию хранилищ баз данных, имеющих существенные отличия от моделей, используемых в традиционных реляционных СУБД (РСУБД) с доступом к данным средствами языка SQL.

Для NoSQL обычно приводятся следующие преимущества по отношению к РСУБД [2].

1. Горизонтальное масштабирование. В NoSQL базах данных эта процедура обычно проще и прозрачнее, чем в РСУБД.

2. Производительность БД на одном узле, а не в кластере, также является немаловажным параметром.

3. Надежная работа в условиях, когда отказ аппаратного обеспечения или сетевая недоступность – обычное дело, является одним из свойств многих решений NoSQL [3]. Основной способ ее обеспечения – это репликация.

4. Простота разработки и администрирования – также важный аргумент в пользу NoSQL-технологий. Целый ряд задач, связанных с масштабированием и репликацией, представляющих значительную сложность и требующих обширной специальной экспертизы на традиционных СУБД, у NoSQL занимает считанные минуты. Задачи установки и настройки, само использование NoSQL-решений обычно существенно проще и менее трудоемки, чем в случае с РСУБД.

Важным этапом в исследовании распределенных систем является формулирование Э. Брюером CAP теоремы [4], которая утверждает следующее: из трех свойств распределенной системы:

согласованности (Consistency),

доступности (Availability),

устойчивости к разделению (Partition tolerance),

распределенная система может иметь максимум два одновременно.

Реляционные СУБД основываются на наборе требований ACID, тогда как NoSQL – на наборе требований BASE, при этом реляционная СУБД выносит на первое место требование согласованности, тогда как NoSQL жертвует им в пользу доступности и устойчивости к разделению (согласно CAP теореме) [5].

Согласованность данных обеспечивается за счет поддержки ограничений целостности.

Таким образом, спецификация ограничений целостности для NoSQL баз данных позволит задавать ограничения целостности и частично обеспечить свойство согласованности.

Целью настоящей статьи является спецификация ограничений целостности основных моделей NoSQL баз данных на формальном языке.

Обзор основных моделей NoSQL баз данных

Документно-ориентированные СУБД. Эти СУБД хранят данные в виде коллекций документов, состоящих из набора полей. Этот набор может различаться в документах одной коллекции благодаря «бессхемности» таких СУБД. Идеальный вариант их применения – это хранение более-менее независимых документов, не требующих поддержания ссылочной целостности между ними или коллекциями (форумы или социальные сети, каталоги товаров или изделий). Самые популярные представители этого семейства – MongoDB и CouchDB.

Хранилища типа «ключ-значение». Такие БД хранят данные в виде пар ключ-значение. В некоторых случаях значениями могут быть массивы, списки, множества (наборы уникальных значений) и т.п. Обычно они реализуют минимальный набор операций (установить, прочитать значение и др.).

Типичное применение этих решений – кэширование данных для повышения общей производительности приложения (например, результатов запросов к более сложным системам) и счетчики. Иногда их применяют в качестве промежуточного звена для систем логирования или сбора статистики.

Широко распространенные представители семейства «ключ-значение» – Redis и Riak .

Колоночные СУБД. В отличие от традиционных БД, колоночные СУБД хранят данные в виде последовательности столбцов, а не строк. Благодаря этому достигаются некоторые преимущества в хранении и обработке больших объемов информации.

Типичные задачи, которые решают с помощью колоночных СУБД, – те, для которых скорость (многопоточной) записи обычно важнее скорости чтения (хранение и архивирование данных, логирование и сбор статистики). Подобные СУБД лучше подходят для агрегации больших объемов данных, чем для онлайн-обработки сложных запросов. Один из самых известных представителей этого семейства – BigTable и Apache Cassandra.

Семейство столбцов – это строка, содержащая множество столбцов, ассоциированных с ключом строки.

Семейства столбцов группируют взаимосвязанные данные, доступ к которым часто обеспечивается как к единому целому. Основной единицей хранения в колоночных базах данных является столбец, состоящий из пары «имя-значение», в которой имя играет роль ключа. Каждая из пар «имя-значение» всегда хранится с меткой времени, которая используется для того, чтобы задавать срок действия данных, разрешать конфликты записи, обрабатывать устаревшие данные и выполнять другие функции. Если данные столбца больше не используются, то это место можно восстановить позднее на этапе уплотнения.

Граф-ориентированные СУБД. Такие СУБД эффективно хранят данные, представленные в виде графа: с вершинами (узлами) и ребрами (связями между ними). Идеальны для хранения отношений между множеством сущностей и анализа их взаимосвязей (например, социальный граф, зависимости между компонентами систем и т.п.).

Особой известности граф-ориентированные СУБД не получили, в основном из-за узкого круга решаемых задач.

Спецификация основных моделей NoSQL на формальном языке

В настоящей статье будут рассматриваться две основные NoSQL модели: хранилище типа «ключ значение» на примере Redis и колоночные СУБД на примере BigTable.

Для остальных модели целесообразность спецификации ограничений целостности отсутствует в виду специфики модели и задач, для решения которых они применяются.

В качестве формального модели NoSQL была выбрана теория предсхем и формальный язык описания предсхем – SDL нотация [6 – 7].

Основные элементы предсхем:

Концепт - сущность предметной области.

Роль – ссылка внутри экземпляра понятия на его структурные части.

Квалификатор – это частичное отображен из множества ролей во множество понятий предметной области.

Базовый концепт – концепт, который не определяется с помощью других концептов, т.е. он является атомарным.

Основные элементы **SDL** нотации**:**

item – описывает элемент модели (концепт, квалификатор, шаблон и т.д.);

definition – для описания квалификаторов, связанных с концептом.

selector – для описания пары (роль-концепт);

define –ключевое слово для определения сложного квалификатора;

template – ключевое слово для определения шаблонного концепта (например List<X>, Set<X> и т.д.);

typedef – ключевое слово для определения сокращенного имени, например для шаблонного концепта (например typedef List<User> = ListUsers);

is_atomic – ключевое слово для определения базового концепта.

Хранилища типа «Ключ-Значение» (Redis).

База данных типа «Ключ-Значение» – это список пар (ключ, значение). В качестве значений ключа выступают только строки, а значение могут быть

одним из следующих типов, приведенных в табл. 1. Также в табл. 1 приведено отображения типов данных Redis в типы данных, соответствующие предсхеме хранилища «Ключ-Значение».

Таблица 1

Отображение типов данных Redis

Тип данных в БД	Целевой тип данных
String	String
List	List<String>
Set	List<String>
Bitmap	Bitmap
Hash	List<HashElement>

Приведем описание хранилища «Ключ-Значение» на выбранном формальном языке.

База данных типа «Ключ-Значение» – это список концептов Object.

Ниже приведено формальное описание предсхемы хранилища «Ключ-Значение».

SDL нотация предсхемы хранилища Redis

```
define qObject (KeyName:String, KeyValue:Value).
Object= qObject.
String is_atomic.

define qBitMap (BitOffset:int, BitValue:bit,
Func:String).
BitMap=qBitMap.
int is_atomic.
bit is_atomic.

define qHashEl(KeyName:String, KeyValue:String).
HashElement = qHashEl.

typedef List<String> = Set.
typedef List<String> = List.
typedef List<HashElement> = Hash.

Value = String;List;Set;Bitmap;Hash.
typedef List< Objects > = KeyValueDataBase.
```

Для хранилищ типа «Ключ-Значение» справедливы следующие ограничения целостности:

- UniqueDBConstraint – каждый ключ в базе данных не должен повторяться.
- UniqueSetConstraint – значения в Set должны быть уникальными.
- UniqueHashConstraint – каждый ключ в Hash должен быть уникальным.

Тогда модель хранилища типа «Ключ-Значение» это предсхема KeyValueDataBase с ограничениями {UniqueDBConstraint, UniqueSetConstraint, UniqueHashConstraint}.

Описание соответствующих ограничений на формальном языке будет приведено далее в статье.

Колоночные СУБД (BigTable).

База данных BigTable – это список элементов, состоящих из ключа строки таблицы и одно или нескольких семейств столбцов.

В качестве значений ключа строки таблицы выступают только строки, а значение в столбцах могут быть одним из следующих типов, приведенных в табл. 1. Также в табл. 1 приведено отображения типов данных BigTable в типы данных, соответствующие предсхеме БД BigTable.

Таблица 2

Отображение типов данных BigTable

Тип данных в БД	Целевой тип данных
Integer	Integer
Double	Double
Boolean	Boolean
Text String	String
Byte String	Byte String
Timestamp	Integer

Приведем описание базы данных BigTable на выбранном формальном языке.

База данных BigTable – это список концептов RowKey. Ниже приведено формальное описание предсхемы BigTable.

SDL нотация предсхемы BigTable

```
define qRowKey(RowKeyName:String,
                RowKeyValue:List<ColumnFamily>).
RowKey= qRowKey.
String is_atomic.

define qColumnFamily (ColumnFamilyName:String,
                ColumnFamilyValue:List<Column>).
ColumnFamily = qColumnFamily.
Column = qColumn(ColumnName:String, Col-
umnValue:Value, TimestampColumn:Integer).
Value = String; Integer; Double; Boolean; ByteString.
Integer is_atomic.
Double is_atomic.
Boolean is_atomic.
ByteString is_atomic.

typedef List< RowKey > = ColumnDataBase.
```

Для базы данных BigTable справедливо следующеие ограничения целостности:

- UniqueRowKeyConstraint – каждый ключ стоки в базе данных должен быть уникальным.

Тогда базы данных BigTable это предсхема ColumnDataBase с одним ограничением – UniqueRowKeyConstraint.

Описание соответствующего ограничения на формальном языке будет приведено далее в статье.

Спецификация ограничений целостности для NoSQL

В качестве формального языка описания ограничений целостности для NoSQL в настоящей статье был использован язык описания структурных ограничений для предсхем, основанный на пути в ациклическом графе.

С точки зрения хранилищ данных, предсхему можно рассматривать как схему (модель) данных, тогда образец концепта – как структуру данных, соответствующую схеме данных.

Образец концепта может быть представлен в виде ациклического графа, а ограничения целостности – с помощью выражений, основывающихся на пути в ациклическом графе [8].

Образец концепта представляется в виде ациклического графа, вершины которого соответствуют либо выбранным квалификаторам, либо базовым концептам. Ребра графа помечаются именами ролей в рамках определения квалификатора.

На рис. 1 приведен пример предсхемы концепта «List», а на рис. 2 – список из 2-х целых чисел (образец концепта «List»).

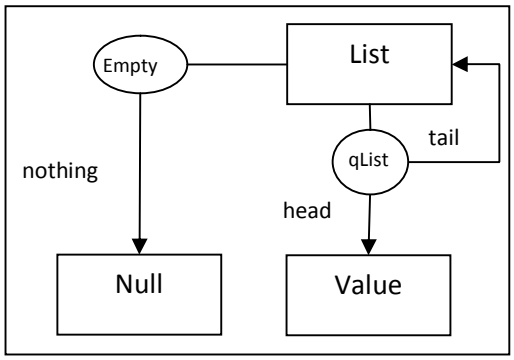

Рис. 1. Предсхема концепта «List»

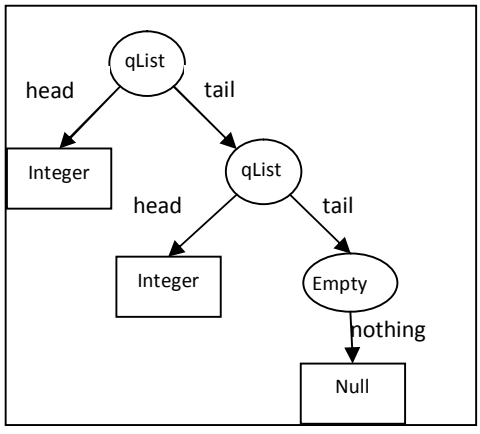

Рис. 2. Образец концепта «List»

Путь в ациклическом графе имеет следующий вид:

идентификатор_концепта.идентификатор_роли: идентификатор_концепта.идентификатор_роли

где "**:**" (двоеточие) – выбор концепта, "**.**" (точка) – выбор роли.

Тогда ограничения целостности – это выражения на основе путей в графе и, возможно, кванторов существования и всеобщности.

Зададим ограничения целостности для NoSQL, описанные выше.

Для хранилищ типа «Ключ-Значение» были определены следующие ограничения целостности:

- UniqueDBConstraint – каждый ключ в базе данных не должен повторяться.

  UniqueDBConstraint :=
  ∀ qList.head:qObject.KeyName !=
  qList.tail:qList.head: qObject.KeyName

- UniqueSetConstraint – значения в Set должны быть уникальными.

  UniqueSetConstraint:=
  ∀ qList.head != qList.tail: qList.head

- UniqueHashConstraint – каждый ключ в Hash должен быть уникальным.

  UniqueHashConstraint:=
  ∀ qList.head:qHashEl.KeyName!=
  qList.tail: qList.head:qHashEl.KeyName

Для базы данных BigTable были определены следующие ограничения целостности:

- UniqueRowKeyConstraint – каждый ключ стоки в базе данных должен быть уникальным.

  UniqueRowKeyConstraint:=
  ∀ qList.head:qRowKey.RowKeyName!=
  qList.tail:qList.head:
  qRowKey.RowKeyName

Таким образом, описаны на формальном языке ограничения, определенные в каждом из рассмотренных типов хранилищ. Подобным образом могут быть описаны ограничение, заданные пользователем для каждой из моделей.

Алгоритм обеспечения свойства согласованности для NoSQL

Основой предлагаемого метода обеспечения свойства согласованности для NoSQL является модуль Checker, который проверяет соответствие изменений, вносимых в БД, тем ограничениям, которые заданы. Схема алгоритма проверки ограничений целостности с помощью модуля Checker представлена на рис. 3.

Согласно приведенной схеме, модуль Bridge фильтрует запросы к БД – запросы-выборки напрямую обращаются к БД, а запросы-модификаторы перенаправляются на модуль Checker для проверки допустимости вносимых ограничений. Таким образом, не нужно вносить никаких изменений в суще-

ствующие NoSQL СУБД, а достаточно все запросы пропускать через модуль-фильтр Bridge.

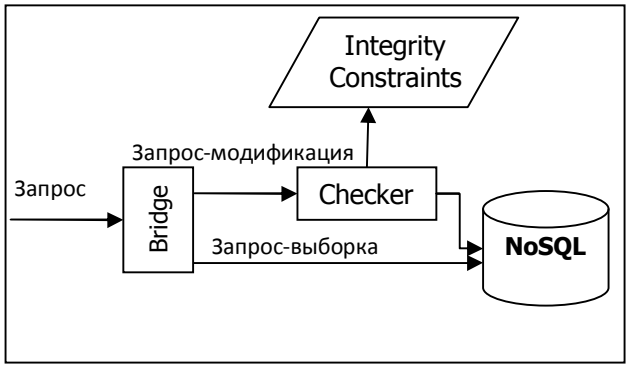

Рис. 3. Схема алгоритма проверки ограничений целостности

Заключение

В работе рассмотрены основные модели NoSQL баз данных, а именно графовые, колоночные, документно-ориентированные и хранилища вида «ключ-значение».

Рассмотрена целесообразность описание ограничений целостности для каждой из описанных моделей.

Ввиду особенностей построения и сфер использования, было принято решение рассматривать только две модели – колоночные (на примере BigTable) и хранилища типа «ключ-значение» (на примере Redis).

Каждая из NoSQL моделей была описана на формальном языке, в качестве которого была выбрана SDL нотация.

Это позволило на формальном языке, основанном на пути в ациклическом графе, специфицировать ограничения целостности. Также, был предложен алгоритм проверки ограничений целостности с помощью модуля Checker.

Таким образом, в работе был предложен метод описания структурных ограничений для NoSQL баз данных, а также алгоритм проверки ограничений целостности с помощью модуля Checker.

Это позволит задавать пользовательские ограничения целостности для NoSQL баз данных, а также верифицировать изменения, вносимы в данные, с помощью модуля Checker.

Список литературы

1. Özsu M.T. Principles of Distributed Database Systems; 3rd edition / M.T. Özsu, P. Valduriez. – Springer, 2011.

2. Фаулер М. NoSQL: новая методология разработки нереляционных баз данных / М. Фаулер, П.Д. Садаладж. – М.: «Вильямс», 2013. – 192 с.

3. Brewer Eric A. Towards robust distributed systems / Eric A. Brewer // Proceedings of the XIX annual ACM symposium on Principles of distributed computing. – Portland, OR: ACM, 2000. – Т. 19. – № 7.

4. Carter A. The CAP Theorem as it Applies to Contemporary NoSQL Storage Systems / A. Carter. – Memorial University, 2011.

5. Кузнецов С. Транзакционные параллельные СУБД: новая волна / С. Кузнецов. 2010. [Электронный ресурс]. – Режим доступа к статье: http://citforum.ru/database/articles/kuz_oltp_2010/2.shtml.

6. Zhytaruk A.G. About language for data structures modeling / Zhytaruk A.G., Zholtkevych G.N. // Системи обробки інформації. – Х.: ХУ ПС, 2011. – Вип. 5(95). – С. 197-201.

7. Житарюк А.Г. Трансляция и верификация SDL–нотации средствами языка ПРОЛОГ / А.Г. Житарюк, Г.Н. Жолткевич // Вісник Херсонського національного університету. – 2009. – Вип. 2. – С. 200-208.

8. Житарюк А.Г. Представление образцов концептов информационных систем посредством ациклических графов / А.Г. Житарюк, Г.Н. Жолткевич // Системи обробки інформації. – Х.: ХУ ПС, 2010. – Вип. 6(87). – С. 215-219.

Поступила в редколлегию 10.09.2015

Рецензент: д-р техн. наук, проф. В.А. Филатов, Харьковский национальный университет радиоэлектроники, Харьков.

СПЕЦИФІКАЦІЯ ОБМЕЖЕНЬ ЦІЛІСНОСТІ ДЛЯ NOSQL БАЗ ДАНИХ

А.Г. Морозова, Л.П. Бєлова, В.О. Погребняк

В роботі запропоновано метод опису обмежень цілісності для NoSQL баз даних. Розглянуто основні моделі NoSQL баз даних, а також доцільність опису обмежень цілісності для кожної з моделей. Для опису обмежень цілісності було запропоновано описати кожну NoSQL модель на формальній мові - SDL в рамках теорії предсхем, що дозволило сформулювати обмеження цілісності на формальній мові, яка базується на понятті шляху в ациклічному графі. В рамках цього підходу розроблено алгоритм перевірки обмежень цілісності для основних моделей NoSQL баз даних.

Ключові слова: *бази даних, моделі даних, NoSQL, розподілені бази даних, горизонтальне масштабування, сховище ключ-значення, документно-орієнтовані бази даних, сховища сімейств колонок, графові бази даних.*

CONSISTENCY SPECIFICATION FOR NOSQL DATABASES

A.G. Morozova, L.P. Belova, V.A. Pogrebniak

The method for describing constraints for NoSQL databases was considered. The basic model of NoSQL databases and advisability of describing constraints for each of the models have been considered. It was proposed to describe every NoSQL model in a formal language - SDL in the theory of preschemes that allowed formulating constraints in a formal language, based on the way to an acyclic graph. On the basis of the method the algorithm for checking NoSQL integrity constraints has been developed.

Keywords: *databases, data models, NoSQL, distributed databases, scaling, key-value stores, document-oriented database, storage columns families, graph database.*